Wpf expander header template binding

Contact us Privacy Policy Cookie Policy Misc. Controls: The Expander control will give you the ability to hide/show content. This would usually be a piece of text, but thanks to wpf's flexibility, it can be used for any type of mixed content such as text, images, and even other WPF controls. To see what I'm talking about, here's an example:Note the arrow part - as soon as you click on it, the Expander control will expand and reveal its contents:The code for this is obviously very simple:<Expander> <TextBlock textwrapping=Wrap fontsize=18>here we can have text that can be hidden/shown using the built-in functionality of the Expander control.</TextBlock> </Expander> By default, the Expander is NOT expanded and therefore looks like it does so on the first screenshot. The user can expand it by clicking it, or you can expand it initially by using the IsExpanded property: you can of course also read this property at run time, if it is<Expander isexpanded=True>you need to know the current state of the Expander control. Advanced content The contents of the Expander can only be one control, as in our first example where we use a TextBlock control, but nothing prevents you from doing so, such as a panel, which can then contain as many child controls as you want. This allows you to host rich content within the Expander, from text and images to a ListView control or any other WPF control. Here's an example of more advanced content, where we use different panels, text, and an image, and even a TextBox control:<Expander margin=10> <StackPanel margin=10><DockPanel><Image source=/WpfTutorialSamples;component/Images/question32.png width=32 height=32 dockpanel.dock=Right margin=10></Image><TextBlock textwrapping=Wrap fontsize=18>Did you know WPF is really great? Just enter your email address below and we'll send you updates:</TextBlock> </DockPanel> <TextBox margin=10>john@doe.org</TextBox> </StackPanel> </Expander>ExpandDirection By default, the Expander control will take up all available space within its container control, and then expand by the value of the ExpandDirection property, which is set to Down by default. You can see it shown in the screens above because the arrow is positioned above the control and points up or down depending on whether the control has been expanded or not. Changing the value of the ExpandDirection property will affect the appearance and appearance of the Expander control. For example, if you change the value to Right, the arrow is placed on the left side and points left/right instead of up/down. Here's a margin=10 expanddirection=Right> <TextBlock textwrapping=Wrap fontsize=18>here we can have text that can be hidden/shown using the built-in functionality of the Expander control.</TextBlock> </Expander> Of course you can also set this property to Up or Left - if you do, the button will be positioned at the bottom or for</Expander> for</Expander> To the right. Custom header In all examples so far, the Expander control is almost appearanceless, except for the button used to show/hide content: it is drawn as a circular button with an arrow inside. However, you can easily customize the header area of the control by using the Header property. Here's an example where we use this property to add explanatory text next to the button: here we can have text that<Expander margin=10 header=Click to show/hide content... > <TextBlock textwrapping=Wrap fontsize=18>can be hidden/shown using the expander control's built-in functionality.</TextBlock> </Expander> But you don't have to settle for a simple piece of text: the Header property will allow you to add controls to it, to create an even more personalized look:<Expander margin=10> <Expander.Header><DockPanel verticalalignment=Stretch><Image source=/WpfTutorialSamples;component/Images/bullet_green.png height=16 dockpanel.dock=Left></Image><TextBlock fontstyle=Italic foreground=Green>Click to show/hide content... </TextBlock> </DockPanel></Expander.Header> <TextBlock textwrapping=Wrap fontsize=18>Here we can have text that can be hidden / shown using the integrated functionality of the Expander control.</TextBlock> </Expander> Note how I simply add a panel as the content of the Header property and within it, I can do what I want, how to add an image and textblock control with custom formatting:Summary The Expander control is a great little help when you need the ability to hide/show content on demand, and just like any other control in the WPF framework, it's both easy to use and easy to customize. This article has been fully translated into the following languages: Chinese French German Italian Polish Spanish Vietnamese Your preferred language is not on the list? Click here to help us translate this article into your own language! Whenever you customize an Expander in WPF using a HeaderTemplate, make a critical error. I forget to set the binding for the header. Here is an artificial example to demonstrate the problem and the solution. That's what we're aiming for. A simple Expander with a title and a few lines of text contained inside. Of course, a HeaderTemplate is excessive here, but it is necessary to prove the problem. Let's start by creating a simple view model that the Expander needs to bind to: DemoViewModel public class { public string Title { get; set; } public string ContentLine1 { get; set; } public string ContentLine2 { get; set; } public string ContentLine3 { get; set; } } now create an instance of the e impostarla come contesto dati per la finestra: public partial class MainWindow : Window { public DemoViewModel ViewModel { get; set; } public MainWindow() { InitializeComponent(); InitializeViewModel(); } void privato InitializeViewModel() { ViewModel = new DemoViewModel { Title = Expander Title, ContentLine1 = This is line 1, ContentLine2 = This is line 2, ContentLine3 = = is line 3 }; This. DataContext = ViewModel; } } Switch to XAML mode and create an Expander with a HeaderTemplate: <Expander width=200> <Expander.HeaderTemplate> <DataTemplate> <StackPanel orientation=Horizontal> <TextBlock text={Binding Title}></TextBlock> </StackPanel> </DataTemplate> </Expander.HeaderTemplate> <StackPanel> <TextBlock text={Binding ContentLine1}></TextBlock> <TextBlock text={Binding ContentLine2}></TextBlock> <TextBlock text={Binding ContentLine3}></TextBlock> </StackPanel> </Expander> when you run this code, you'll notice that the header is empty. Bindings within HeaderTemplate do not bind. I'm not an expert, but this seems to be due to the fact that the data context for the header is not inherited from the expander data context... which seems a little strange to me, but I'm sure there's a good reason. The fix is quite simple, just add the Header={Binding} attribute to the Expander: <Expander header={Binding} width=200>... Bindings within HeaderTemplate should now work: this entry was published to reference and marked with C#, WPF, XAML. Bookmark the permalink. When creating a simple custom expander, I encountered the problem where items within IT didn't associate. I found the fix on this link: t-forget-the-binding/ That happens to deal with exactly the same problem, however what I understand from it is found this thankfully, I'm not really sure why it worked ;D My question is now: why adding Header={Binding} solves the problem. In fact from the fact that the binding will not work, it seems that it is due to the DataContext, but I do not see how this should fix it. Thank you for explaining; let's hope this isn't a < Binding Data to the Header and Content Panel Using TemplatesIn this topic, you will learn how to bind data to the C1Expander control's heading and content panel using the ContentTemplate template and HeaderTemplate template. This topic assumes that you are working in Microsoft Expression Blend. Step 1: Add the C1Expander Control to the Project and Prepare it for Data BindingComplete the following steps:1. Add a C1Expander control to your WPF project.2. Select the C1Expander control to expose its properties in the Properties tab and complete the following:3. Set the Name property to NameAgeHolder1. 4. Next to the Content property, click the Advanced options button and select Custom Expression. Set the Custom expression field to {Binding}. This sets up the Content property to pass the DataContext directly to its template, which you will create in a later step.5. Next to the Header property, click the options button and select Custom Expression. Set the Custom expression field to {Binding}. This sets up the Header to pass the DataContext directly to its template, which you will create in a later step. Step 2: Create Templates, Add a Control to Each Template, and Bind Each Control to a Data Source PropertyComplete the following steps:1. Create the first template, the binding= data= to= the= header= and= content= panel= using= templatesin= this= topic,= you= will= learn= how= to= bind= data= to= the= c1expander= control's= heading= and= content= panel= using= the= contenttemplate= template= and= headertemplate= template.= this= topic= assumes= that= you= are= working= in= microsoft= expression= blend.= step= 1:= add= the= c1expander= control= to= the= project= and= prepare= it= for= data= bindingcomplete= the= following= steps:1.= add= a= c1expander= control= to= your= wpf= project.2.= select= the= c1expander= control= to= expose= its= properties= in= the= properties= tab= and= complete= the= following:3.= set= the= name= property= to= nameageholder1.= 4.= next= to= the= content= property,= click= the= advanced= options= button= and= select= custom= expression.= set= the= custom= expression= field= to= {binding}.= this= sets= up= the= content= property= to= pass= the= datacontext= directly= to= its= template ,= which= you= will= create= in= a= later= step.5.= next= to= the= header= property,= click= the= advanced= options= button= and= select= custom= expression.= set= the= custom= expression= field= to= {binding}.= this= sets= up= the= content= property= to= pass= the= datacontext= directly= to= its= template,= which= you= will= create= in= a= later= step.step= 2:= create= templates,= add= a= control= to= each= template,= and= bind= each= control= to= a= data= source= propertycomplete= the= following= steps:1.= create= the= first= template,= the=></ Binding Data to the Header and Content Panel Using TemplatesIn this topic, you will learn how to bind data to the C1Expander control's heading and content panel using the ContentTemplate template and HeaderTemplate template. This topic assumes that you are working in Microsoft Expression Blend. Step 1: Add the C1Expander Control to the Project and Prepare it for Data BindingComplete the following steps:1. Add a C1Expander control to your WPF project.2. Select the C1Expander control to expose its properties in the Properties tab and complete the following:3. Set the Name property to NameAgeHolder1. 4. Next to the Content property, click the Advanced options button and select Custom Expression. Set the Custom expression field to {Binding}. This sets up the Content property to pass the DataContext directly to its template, which you will create in a later step.5. Next to the Header property, click the Advanced options button and select Custom Expression. Set the Custom expression field to {Binding}. This sets up the Header to pass the DataContext directly to its template, which you will create in a later 2: Create Templates, Add a Control to Each Template, and Bind Each Control to a Data Source PropertyComplete the following steps:1. Create the first template, the > duplicato >.</Expander> >.</Expander> completing the following steps:a. Click the NameAgeHolder1 location and select Edit Additional Templates | Edit HeaderTemplate | Create blank. The Create DataTemplate Resource dialog box opens. B. In the Name (key) field, enter NameTemplate. c. Click OK to close the Create DataTemplate Resource dialog box to create the new template. q. Click the Resources tab, and then, in the search field, enter Label to find the Label.e control. Double-click the Label icon to add the Label control to the template. F. Select the Label control to expose its properties on the Properties tab. G. Next to the Content property, click the Advanced Options button and select Custom Expression. Set the custom expression field to {Binding Name}. This sets the Content property of the Label control to the value of the Name property, which is a property that will be created in code later. 2. Create the second template, ContentTemplate, by completing the following steps:a. Click the NameAgeHolder1 location and select Edit Additional Templates | Edit Generated Content (ContentTemplate) | Create blank. The Create DataTemplate Resource dialog box opens. B. In the Name (Key) field, enter AgeTemplate. c. Click OK to close the Create DataTemplate Resource dialog box to create the new template. q. Click the Resources tab, and then, in the search field, enter Label to find the Label.e control. Double-click the Label icon to add the Label control to the template. F. Select the Label control to expose its properties on the Properties tab. G. Next to the Content property, click the Advanced Options button and select Custom Expression. Set the Custom Expression field to {Binding Age}. This sets the Content property of the Label control to the value of the Age property, which is a property that will be created in code later. Step 3: Create Data Source Completion in the following steps:1. Open the MainPage.xaml code page (it will MainPage.xaml.cs or MainPage.xaml.vb depending on the language you choose for your project. 2. Add the following class to the project, placing it under the namespace declaration:• Visual BasicPublic Class NameAndAge Public Sub New(name As String, age As Integer) Name = name Age = age End Sub Public Property Name() As String Get End Get End Set End Property Age() As Integer Get End Get Set End Set End PropertyEnd Class• C#public class NameAndAge { public NameAndAge(string name, int age) { Name = Age = age; } public string Name { get; set; } public int Age { get; set; } } This class creates a class with two properties: a string property named Name and a numeric property named Age. 3. the following code under the InitializeComponent() method to set the Name property and the Age property:• Visual BasicNameAgeHolder1.DataContext = New NameAndAge(Gaius Baltar, 40)• C#NameAgeHolder1.DataContext = new NameAndAge(Gaius Baltar, 40); Step 4: Run the project and observe resultsComplete in the following steps:1. press f5 to run the Note that the value of the Name property appears on the control's header bar.2. Click the header bar to expand the control and see that the Value of the Age property appears in the content panel: